Tutorial: Modeling Izhikevich Neuron

This tutorial demonstrates how to model the Izhikevich neuron using Verilog HDL and Python. It includes a testbench, waveform generation, and Python visualization.

---

1. Introduction

- **Izhikevich Model**: A computationally efficient neuron model balancing biological realism and computational simplicity. More detail is on this page https://www.izhikevich.org/publications/spikes.htm
- **Equations**:
  - $v' = 0.04v^2 + 5v + 140 - u + I$
  - $u' = a(bv - u)$
  - Spike condition: when $v \geq 30$, reset $v = c$ and $u = u + d$.
- **Applications**: Neural modeling, spiking neural networks, neuromorphic systems.

---

2. Mathematical Formulation

- **State Variables**:
  - v: Membrane potential.
  - u: Recovery variable.
- **Parameters**:
  - a, b, c, d: Define neuron behavior (e.g., regular spiking, bursting).

---

3. Python Implementation

Code for Simulation and Visualization

```
import numpy as np
import matplotlib.pyplot as plt

# Parameters
time_steps = 1000
dt = 0.5
a, b, c, d = 0.02, 0.2, -65, 8
v = -65
u = b * v
threshold = 30
```

```python
input_current = np.zeros(time_steps)
input_current[100:800] = 10  # Step input

# Storage for plotting
v_trace = []
u_trace = []
spikes = []

# Simulation
for t in range(time_steps):
    v_trace.append(v)
    u_trace.append(u)
    if v >= threshold:
        v = c
        u += d
        spikes.append(1)
    else:
        spikes.append(0)
        dv = 0.04 * v**2 + 5 * v + 140 - u + input_current[t]
        du = a * (b * v - u)
        v += dv * dt
        u += du * dt

# Plot results
plt.figure(figsize=(12, 6))
plt.subplot(2, 1, 1)
plt.plot(v_trace, label="Membrane Potential (v)")
plt.axhline(y=threshold, color="r", linestyle="--", label="Threshold")
plt.legend()
plt.subplot(2, 1, 2)
plt.stem(spikes, linefmt="r-", markerfmt="ro", basefmt="k-", label="Spikes")
plt.legend()
plt.show()
```

Illustration

The plot of the Izhikevich neuron is shown below:

---

4. Verilog HDL Implementation

Izhikevich Neuron Code

```verilog
module Izhikevich_Neuron (
    input clk,
    input reset,
```

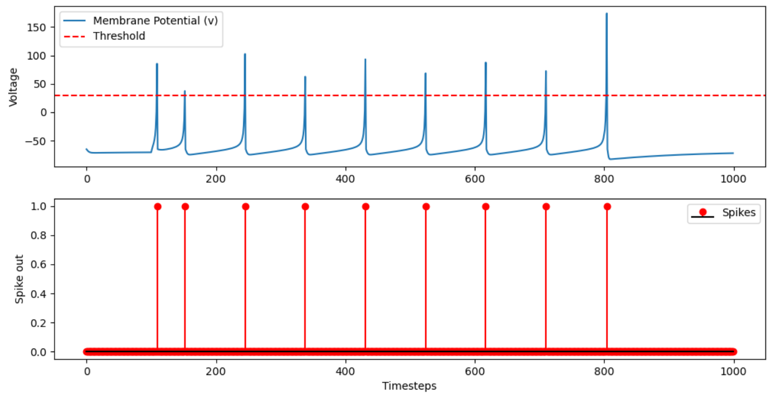


Figure 1: Waveform of Izhikevich neuron

```
    input signed [15:0] input_current,
    output reg signed [15:0] v, // Membrane potential
    output reg spike
);
    // Parameters
    parameter signed [15:0] a = 16'sd2;   // Recovery time scale
    parameter signed [15:0] b = 16'sd2;   // Sensitivity of u
    parameter signed [15:0] c = -16'sd65; // Reset value for v
    parameter signed [15:0] d = 16'sd8;   // Reset increment for u
    parameter signed [15:0] v_threshold = 16'sd30;

    // Internal variables
    reg signed [15:0] u; // Recovery variable

    // Update logic
    always @(posedge clk or posedge reset) begin
        if (reset) begin
            v <= -16'sd70; // Initial membrane potential
            u <= 16'sd0;   // Initial recovery variable
            spike <= 1'b0;
        end else begin
            // Spike condition
            if (v >= v_threshold) begin
                v <= c;          // Reset membrane potential
                u <= u + d;      // Increment recovery variable
                spike <= 1'b1;   // Output spike
            end else begin
```

```
                spike <= 1'b0;
                // Update equations
                v <= v + ((16'sd4 * v * v) / 16'sd10) + (16'sd5 * v) + 16'sd140 - u + input_
                u <= u + a * (b * v - u);
            end
        end
    end
endmodule
```

Testbench

Testbench Code

```
`timescale 1ns/1ps

module Izhikevich_Neuron_tb;

    // Testbench signals
    reg clk;
    reg reset;
    reg signed [15:0] input_current;
    wire signed [15:0] v;
    wire spike;

    // Instantiate the DUT (Device Under Test)
    Izhikevich_Neuron dut (
        .clk(clk),
        .reset(reset),
        .input_current(input_current),
        .v(v),
        .spike(spike)
    );

    // Clock generation
    always #5 clk = ~clk; // 10 ns clock period

    // Testbench sequence
    initial begin
        // Enable waveform generation
        $dumpfile("waveform.vcd");
        $dumpvars(0, Izhikevich_Neuron_tb);

        // Initialize signals
        clk = 0;
        reset = 1;
```

```
        input_current = 0;

        // Apply reset
        #10 reset = 0;

        // Test case 1: Small input current
        input_current = 16'sd5;
        #100;

        // Test case 2: Larger input current
        input_current = 16'sd20;
        #100;

        // Test case 3: Reset during operation
        reset = 1;
        #10 reset = 0;
        #100;

        // End simulation
        $stop;
    end

    // Monitor signals
    initial begin
        $monitor($time, " Reset=%b, Input=%d, v=%d, Spike=%b",
                 reset, input_current, v, spike);
    end

endmodule
```

Simulation and waveform

Before running the simulation, please check out some explanations about iverilog and gtkwave here.

To run the simulation, please follow this instructions.

Install iverilog and gtkwave If your machine does not have these tools, please install them from :

- iverlog: https://steveicarus.github.io/iverilog/
- gtkwave: https://gtkwave.sourceforge.net/

Note: if you are familiar with hardware design, you can use different tools to simulate.

Run the simulation First, check the current work folder (`hw`) and the availability of the tools:

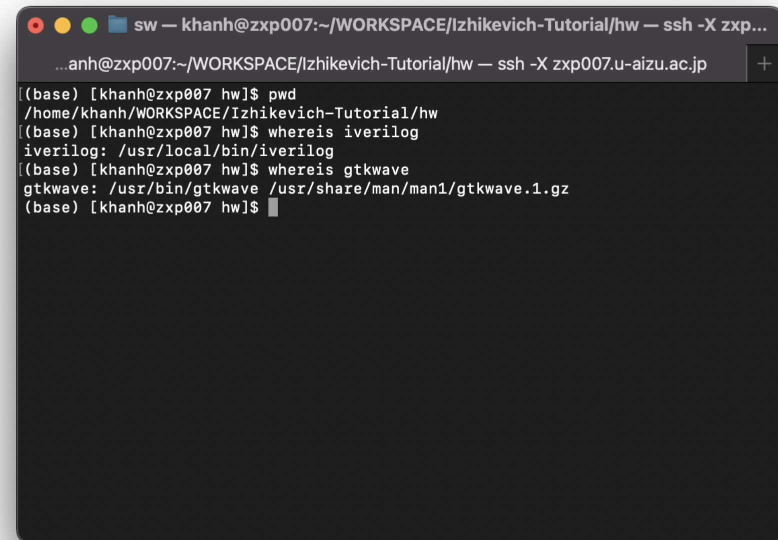
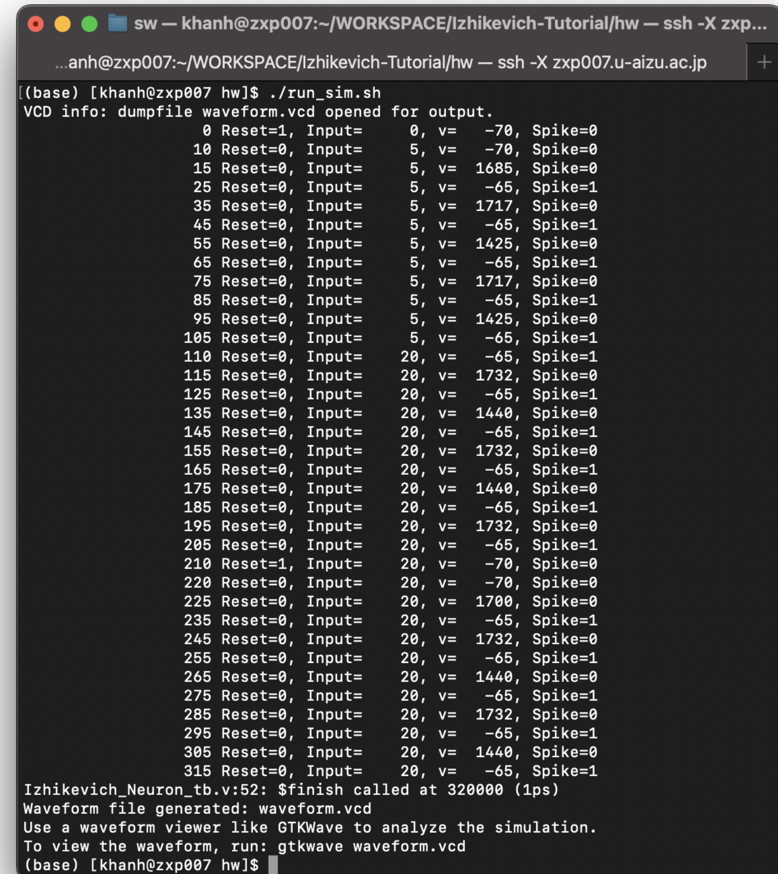


Figure 2: Check the availability of the tools.

Then, run the `run_sim.sh` script:




```
(base) [khanh@zxp007 hw]$ ./run_sim.sh
VCD info: dumpfile waveform.vcd opened for output.
                   0 Reset=1, Input=     0, v=   -70, Spike=0
                  10 Reset=0, Input=     5, v=   -70, Spike=0
                  15 Reset=0, Input=     5, v=  1685, Spike=0
                  25 Reset=0, Input=     5, v=   -65, Spike=1
                  35 Reset=0, Input=     5, v=  1717, Spike=0
                  45 Reset=0, Input=     5, v=   -65, Spike=1
                  55 Reset=0, Input=     5, v=  1425, Spike=0
                  65 Reset=0, Input=     5, v=   -65, Spike=1
                  75 Reset=0, Input=     5, v=  1717, Spike=0
                  85 Reset=0, Input=     5, v=   -65, Spike=1
                  95 Reset=0, Input=     5, v=  1425, Spike=0
                 105 Reset=0, Input=     5, v=   -65, Spike=1
                 110 Reset=0, Input=    20, v=   -65, Spike=1
                 115 Reset=0, Input=    20, v=  1732, Spike=0
                 125 Reset=0, Input=    20, v=   -65, Spike=1
                 135 Reset=0, Input=    20, v=  1440, Spike=0
                 145 Reset=0, Input=    20, v=   -65, Spike=1
                 155 Reset=0, Input=    20, v=  1732, Spike=0
                 165 Reset=0, Input=    20, v=   -65, Spike=1
                 175 Reset=0, Input=    20, v=  1440, Spike=0
                 185 Reset=0, Input=    20, v=   -65, Spike=1
                 195 Reset=0, Input=    20, v=  1732, Spike=0
                 205 Reset=0, Input=    20, v=   -65, Spike=1
                 210 Reset=1, Input=    20, v=   -70, Spike=0
                 220 Reset=0, Input=    20, v=   -70, Spike=0
                 225 Reset=0, Input=    20, v=  1700, Spike=0
                 235 Reset=0, Input=    20, v=   -65, Spike=1
                 245 Reset=0, Input=    20, v=  1732, Spike=0
                 255 Reset=0, Input=    20, v=   -65, Spike=1
                 265 Reset=0, Input=    20, v=  1440, Spike=0
                 275 Reset=0, Input=    20, v=   -65, Spike=1
                 285 Reset=0, Input=    20, v=  1732, Spike=0
                 295 Reset=0, Input=    20, v=   -65, Spike=1
                 305 Reset=0, Input=    20, v=  1440, Spike=0
                 315 Reset=0, Input=    20, v=   -65, Spike=1
Izhikevich_Neuron_tb.v:52: $finish called at 320000 (1ps)
Waveform file generated: waveform.vcd
Use a waveform viewer like GTKWave to analyze the simulation.
To view the waveform, run: gtkwave waveform.vcd
(base) [khanh@zxp007 hw]$
```

Figure 3: Run the simulation script.

Finally, run `gtkwave waveform.vcd` to view the waveform.

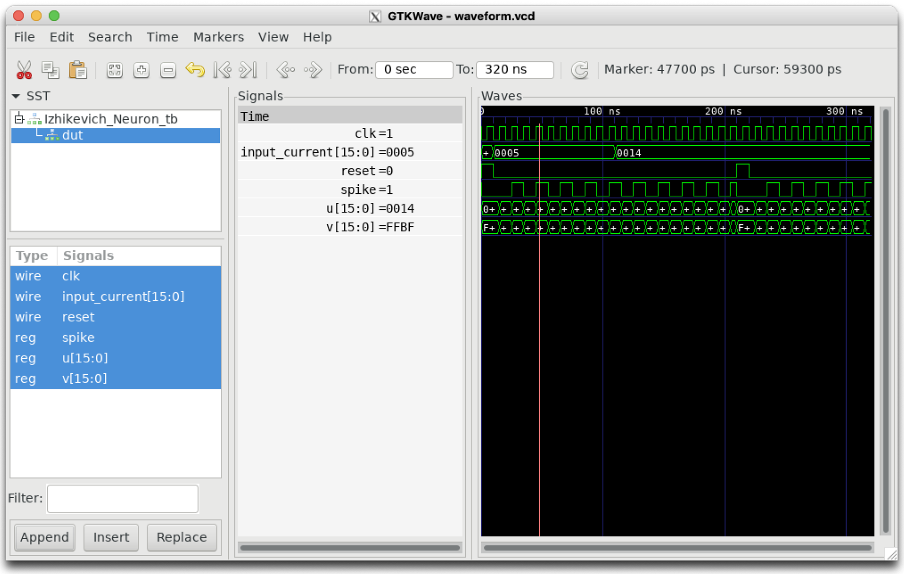


Figure 4: Open the waveform and view the signals.

5. Exercises

- Using the previous design, you need to modify both Verilog and Python codes and make a report to **show the correctness of the Izhikevich model**.
- Update the parameters: a, b, c, and d to validate different behaviors of the Izhikevich neuron. More detail is on this page https://www.izhikevich.org/publications/spikes.htm